Lists of Words

someonesdad1@gmail.com 9 Dec 2012

This document looks at a python script `lookup.py` that can be used to do word searches in word lists. It also discusses getting decent lists of words, something that can be harder than it seems.

I look words up a lot, for three main reasons:

1. Find words that match a regular expression.
2. Look up a word's definition.
3. Find synonyms for the word.

The `lookup.py` script now lets me do all these types of searches.

lookup.py

The `lookup.py` script is my primary tool for performing word look-ups. If you wish to use it, you'll have to find a suitable `grep` command and put a link to the executable into the script. The reason is that it's much faster to use the `grep` command to do the searching of the dictionary files rather than read them into python and process them. In addition, since I use GNU `grep` on my system (under cygwin), I get colored output in my console window; here's an example of looking up words that contain `hea`:

```
ahead
airhead
...
wrongheadedly
wrongheadedness
```

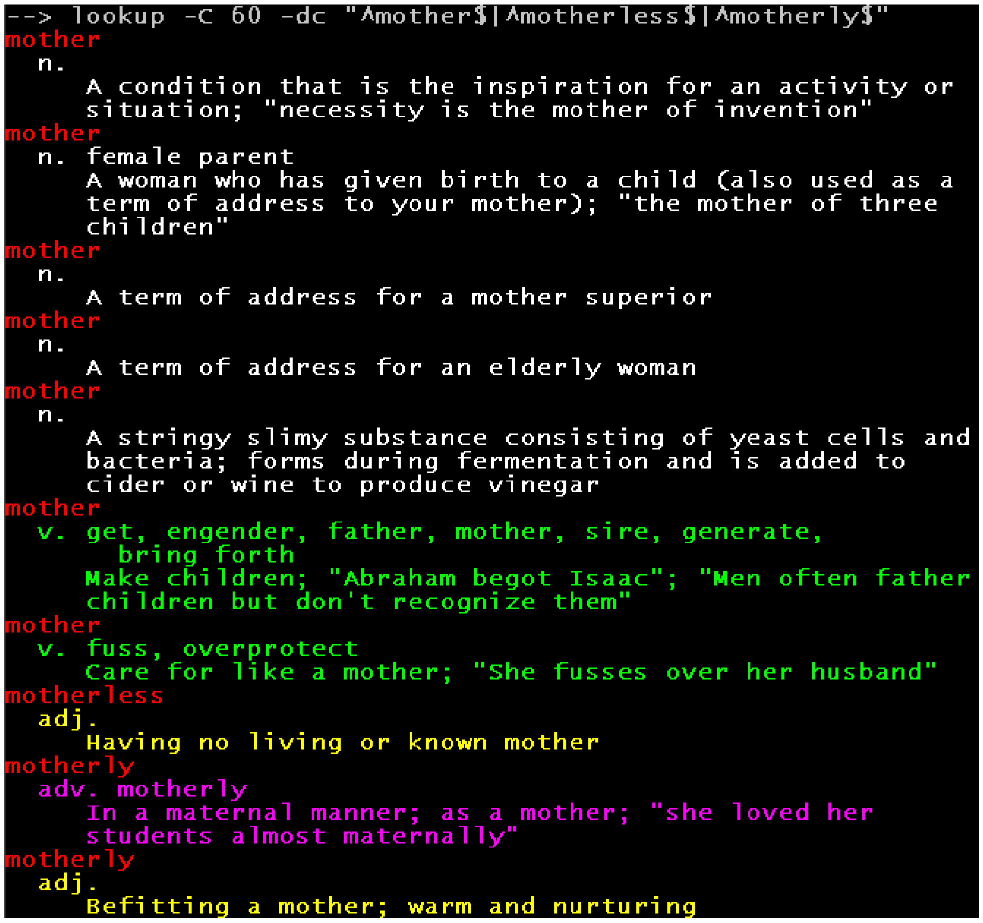
This colorized output is a big time-saver in dealing with lots of output. The majority of the time I'm doing searches where I want to just look up e.g. the word's spelling or find a word that fits a certain regular expression. This tool lets me do this easily and the options let me choose which dictionary to use; here's the usage statement from `lookup.py` (run it without any command line arguments to have the usage statement printed):

```
Usage:  lookup.py [options] regexp
  Look up a regular expression in a dictionary of words.  The search
  tool is grep, so you should use grep's regular expressions.  You'll
  have to make sure the grep variable in the program points to a
  suitable grep program.

  The WordNet option provide the ability to search the list of words
  from WordNet and see synonyms and definitions.  Note that the
  WordNet dictionary also includes combinations of words connected by
  hyphens and space (underscore) characters (use -c to exclude them).

Options:
    -0      Use a simple English dictionary (850 words)
    -1      Use the default dictionary      (42 kwords)
    -2      Use a larger ubuntu dictionary  (98 kwords)
    -3      Use a large dictionary         (274 kwords)
    -C n    Set the number of columns to n
    -w      Use a dictionary from WordNet  (155 kwords)
    -i      Make search case-sensitive

WordNet search options (causes -w option to be set):
    -a      Limit to adjectives
    -c      Do not show compound or hyphenated words
    -d      Show definitions/synonyms for all words that match
    -n      Limit to nouns
    -r      Limit to adverbs
    -v      Limit to verbs

 Acknowledgements:
   1.   Thanks to Alan Beale for his 12dicts (I use his 2of12.txt
        file for my default dictionary):
        http://wordlist.sourceforge.net/12dicts-readme.html.
   2.   Thanks to the folks at Princeton who provide WordNet:
        http://wordnet.princeton.edu/.
```

The added power of `lookup.py` comes from the ability to use the WordNet information once you know the word(s) you're interested in. For example,

```
python lookup.py -C 60 -dc "^mother$|^motherless$|^motherly$"
```

resulted in the colored output (showing the different word types) to my shell window:



```
--> lookup -C 60 -dc "^mother$|^motherless$|^motherly$"
mother
  n.
     A condition that is the inspiration for an activity or
     situation; "necessity is the mother of invention"
mother
  n. female parent
     A woman who has given birth to a child (also used as a
     term of address to your mother); "the mother of three
     children"
mother
  n.
     A term of address for a mother superior
mother
  n.
     A term of address for an elderly woman
mother
  n.
     A stringy slimy substance consisting of yeast cells and
     bacteria; forms during fermentation and is added to
     cider or wine to produce vinegar
mother
  v. get, engender, father, mother, sire, generate,
       bring forth
     Make children; "Abraham begot Isaac"; "Men often father
     children but don't recognize them"
mother
  v. fuss, overprotect
     Care for like a mother; "She fusses over her husband"
motherless
  adj.
     Having no living or known mother
motherly
  adv. motherly
     In a maternal manner; as a mother; "she loved her
     students almost maternally"
motherly
  adj.
     Befitting a mother; warm and nurturing
```

As written, the `lookup.py` script uses a python library `WConio` to generate color output to a DOS/cygwin shell (I work in a cygwin bash window) -- you'll have to get `WConio` if you want colored output in a DOS/cygwin window. If you want to get the same behavior on a UNIX-like system, you'll have to do some simple coding pertinent to your terminal window (change the `color.py` script included in the package). This is often done with escape sequences; one way of finding them is to look at a command line tool like GNU `grep` or `ls` that provide colorized output. Or, you may be able to use e.g. the `ncurses` library. It's also simple to remove the colorizing commands in the `lookup.py` script (they're only in one function), but I find it quite convenient to quickly differentiate the results by color.

If you wish to use the `lookup.py` script with the WordNet features, you'll need to download WordNet 3.0 and construct some files. See WordNet below.

You'll also need to find various word list files to use with the script. I didn't supply any, mainly to save space (see the Getting word lists section for some hints on where to look). You'll have to edit the global variables `dictionary_files` and `dictionary_files` to make sure they point to the right files.

simple.py

The `simple.py` script examines files or stdin for words that are not in Ogden's simple English list of words (851 words). The script changes all non-letter characters to spaces, converts everything to lowercase, then splits the words on space characters. Then the words not in Ogden's list are printed.

This can be a useful tool to ensure that some writing is done at a basic level. However, you'll probably find that few pieces of text will pass this test, so you may find yourself building a somewhat larger dictionary over time.

If you want a more useful tool for assessing the readability of English text, get the `readability.zip` package from http://code.google.com/p/hobbyutil/. Its usage statement is:

```
Usage:  d:/p/pylib/readability.py file1 [file2...]

Prints readability statistics for text files.

FKRE = Flesch-Kincaid Reading Ease
    0-100, higher numbers mean easier to read

The following numbers are the approximate reading level in US grade level:
    FOG  = Gunning Fog Index
    ARI  = Automated Readibility Index
    CL   = Coleman-Liau Index
    FKGL = Flesch-Kincaid Grade Level
    SMOG = SMOG Index
    FORC = FORCAST Readability Formula

See the comments in the program code for formulas and references.
```

When I use this tool, I try to shoot for a grade level of 8 for general writing and keep the grade level at or under 12 for technical writing (it's not always possible, however). Typical output is:

```
  FOG   ARI    CL  FKRE  FKGL  SMOG  FORC
   10     6    14    55     8     9    11  writing.txt
   15    13    10    53    12    13    10  Pride_and_Prejudice.txt
   11     9    10    69     8    10     9  Tom_Sawyer.txt
```

Getting word lists

Finding and/or making good word lists can be a journey rather than the simple task you might think it should be. The easiest thing is to grab a `words` file from a UNIX box, but, as mentioned, these may have more than their fair share of unusual words, giving you lots of matches that you shake your head over. Not that most of them aren't words, but they're "3-sigma" words -- outside (or far outside) common usage. And many of them contain acronyms and abbreviations that really shouldn't be in the list.

Often, what one wants is a selection of dictionaries that can be combined as desired. I'm slowly working towards such a thing, but it's not high on the priority list.

I'm grateful to Alan Beale, who has put together some useful dictionaries at http://wordlist.sourceforge.net/12dicts-readme.html. A quote from Alan's page resonated with me, as it was exactly what I had experienced (the context is the typical UNIX `words` file): "I was not impressed by the overall quality of these lists, and the few which were high-quality were all-inclusive, burying the everyday words under a mountain of archaisms and esoterica." Alan undertook the task of building useful word lists and has apparently spent more than 12 years at it. In particular, I liked his `2of12.txt` file and I now use it as my default dictionary in `lookup.py`.

The `-O` option to the `lookup.py` script is the 850 word list put together by Charles Ogden in 1930 for "basic English". Actually, there are 851 words in the list because it includes grey and gray as acceptable spellings.

Here's a list of 2000 words that may find use as a basic dictionary: http://jbauman.com/aboutgsl.html.

If you do a web search on "english word list", you'll find lots of hits. It will take a lot of time to sort through all the stuff you'll find.

WordNet

Princeton University, "About WordNet", WordNet, Princeton University, 2010, http://wordnet.princeton.edu.

The WordNet project provides a list of words and their meanings and classifications. The files constituting the database are plain ASCII text and can be used as-is, although having programmatic tools for access makes things nicer. I haven't tried the tools that come with WordNet because I just wanted the data so I could use it with the `lookup.py` script, but you might want to examine these tools first to see if they fit your needs better than a console script (there's also a useful web interface for occasional look-ups). If you want to use `lookup.py` with the WordNet files, here are the things you'll need to do:

1. Download the WordNet 3.0 package (see the above link to find it). I downloaded the `WordNet-3.0.tar.bz2` package. Note there's a 3.1 database package available; it contains about 16 words more than the 3.0 package and probably contains updated synonym/definition/linking information.
2. Unpackage the WordNet package and copy the following files from `WordNet-3.0/dict` to a directory where you want them to be stored:

   `data.adj   data.adv   data.noun   data.verb   index.sense`
3. Copy the `index.sense` file to where the `lookup.py` script is installed and rename it; I recommend using the name `words.wordnet`, but you can name it anything you want. Then, using your editor and/or stream tools, remove the regular expression `%.*$` from every line in the `index.sense` file (i.e., delete from the `%` character to the end of the line on each line). Then run the `sort -u` command on the file's contents to sort them in order and keep only unique lines. NOTE: if you don't wish to do this stuff manually, you can run the `mkwords.py` script included in the package and it will make this words file for you. If you have a suitable `make` on your system, edit the `makefile` and let it do all of the requisite work for you.
4. Edit the `lookup.py` script and change the `wordnet_files` global variable to point to these WordNet files. Also edit the `dictionary_files` global variable as appropriate.
5. If you're not running under a DOS or cygwin bash window and you want colored output, you'll have to write substitutes for the functions used from the `color.py` module. The `color.py` and `wrap.py` modules needed are included in the zipfile.
6. Find a suitable `grep` executable and set the `grep` global variable in the script to point to it. If you're on a Windows machine, the `windows/system32/findstr` utility might be able to do what you need (I haven't worked with it or tested it). I'd recommend using the GNU `grep` suite if possible, as it is capable of color-coding the output to show you where the regular expression matched in the output strings (this is useful to sort through lots of output). I haven't tried it, but one such source might be http://unxutils.sourceforge.net/.

Now you should be able to use `lookup.py` with the WordNet features.

If you're a programmer and you wish to make use of WordNet's data files, look at the documentation files, especially `wndb.5.pdf` and `senseidx.5.pdf` for details on the files' structure.